3. Normalform

- Motivation:
  Man möchte zusätzlich verhindern, dass Attribute von nicht-primen Attributen funktional abhängig sind.
- Beispiel:
  LieferAdr (LNr, LName, LStadt, LLand)

001	Huber	München	Deutschland
002	Meier	Berlin	Deutschland
003	Müller	Berlin	Deutschland
004	Hinz	Salzburg	Österreich
005	Kunz	Salzburg	Österreich

- Redundanz: Land mehrfach gespeichert
- Anomalien?

3. Normalform

- Abhängigkeit von Nicht-Schlüssel-Attribut bezeichnet man häufig auch als ***transitive*** (d.h. durch Armstrong-Transitivitätsaxiom hergeleitete) ***Abhängigkeit*** vom Primärschlüssel
  - weil Abhängigkeit *über* ein drittes Attribut besteht:

- Definition:
  Ein Relationenschema ist in 3. Normalform, wenn für jede nicht-triviale funktionale Abhängigkeit $X \rightarrow A$ gilt:
  - X enthält einen Schlüsselkandidaten
  - oder A ist prim.

3. Normalform

Anmerkungen zum Verständnis dieser Definition:

- Wieder Beschränkung auf die FDs, bei denen die rechte Seite nicht-prim ist
  (Abhängigkeiten unter Schlüsselkandidaten sind schwieriger, siehe Boyce-Codd-Normalform)
- Intuitiv möchte die Definition sagen:
  Nicht-prime Attribute sind nur von (ganzen) Schlüsselkandidaten funktional abhängig, also:
  Für jede FD gilt: Linke Seite *ist* ein Schlüsselkandidat.
  - Keine Partiellen FDs von Schlüsselkandidaten:
    → 2. Normalform ist mit 3. Normalform impliziert
  - Keine FDs, bei denen linke Seite kein Schlüssel ist bzw. Teile enthält, die nicht prim sind.

3. Normalform

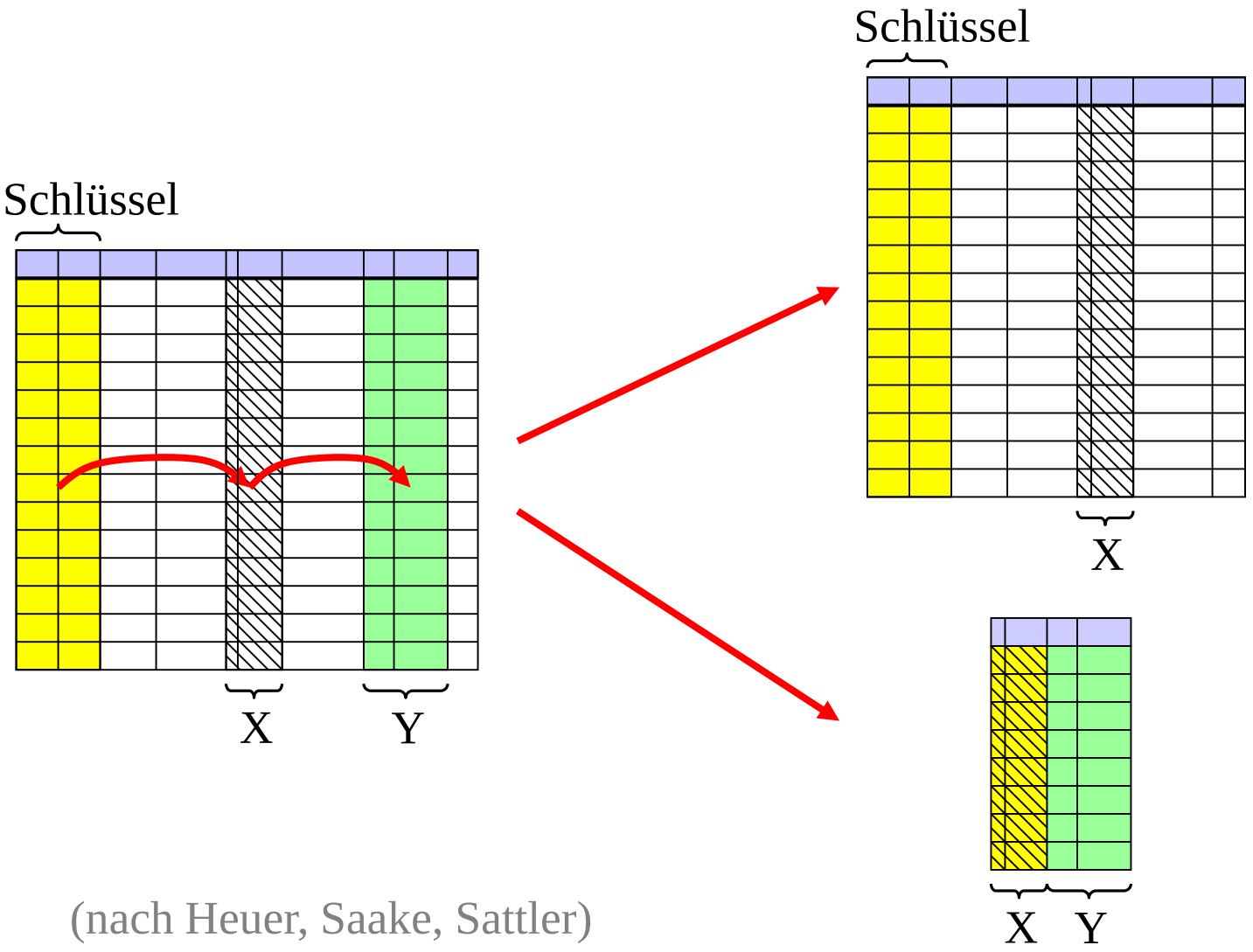
Anmerkungen zum Verständnis dieser Definition:

- Intuitiv möchte die Definition sagen:
  Nicht-prime Attribute sind nur vom (ganzen) Schlüsselkandidaten funktional abhängig, also:
  Für jede FD gilt: Linke Seite *ist* ein Schlüsselkandidat.
- Wegen Reflexivitäts- und Verstärkungsaxiom und Allquantor bei den FDs muss man aber berücksichtigen:
  - Es gelten immer auch die trivialen FDs, z.B. $A \rightarrow A$ (für jedes beliebige Attribut *A*). Deshalb Ergänzung der Definition: „*Für jede nicht-triviale FD gilt…*“
  - Ist *S* Schlüssel, dann gilt immer auch $S' \rightarrow R$ für jede Obermenge $S' \supseteq S$. Deshalb heißt es in der Definition: *X enthält einen Schlüssel*. (statt *X ist ein Schlüssel*.)

3. Normalform

- Transformation in 3. Normalform wie vorher
  - Attribute, die voll funktional abhängig vom Schlüssel sind, und nicht abhängig von Nicht-Schlüssel-Attributen sind, bleiben in der Ursprungsrelation R
  - Für alle Abhängigkeiten $X_i \rightarrow Y_i$ von einem Teil eines Schlüssels ($X_i \subset S$) oder von Nicht-Schlüssel-Attribut:
    - Lösche die Attribute Y_i aus R
    - Gruppiere die Abhängigkeiten nach gleichen linken Seiten X_i
    - Für jede Gruppe führe eine neue Relation ein mit allen enthaltenen Attributen aus X_i und Y_i
    - X_i wird Schlüssel in der neuen Relation

Grafische Darstellung



(nach Heuer, Saake, Sattler)

Verlustlosigkeit der Zerlegung

- Die Zerlegung einer Relation R in die beiden Teilrelationen R_1 und R_2 heißt *verlustlos* (oder auch *verbundtreu*), wenn gilt:

$$R = R_1 \bowtie R_2$$

- Man kann zeigen, dass die Zerlegung verlustlos ist, wenn mindestens eine der beiden folgenden FDs gilt:
  - $(R_1 \cap R_2) \rightarrow R_1$
  - $(R_1 \cap R_2) \rightarrow R_2$
- Intuitiv ist klar, dass Verlustlosigkeit von größter Wichtigkeit ist, weil andernfalls gar nicht dieselbe Information in den Relationenschemata R_1 und R_2 gespeichert werden kann, wie in R.

Abhängigkeitserhaltung

- Die Zerlegung einer Relation R in die beiden Teilrelationen R_1 und R_2 heißt *abhängigkeitserhaltend* (oder auch *hüllentreu*), wenn gilt:

$$F_R = (F_{R_1} \cup F_{R_2}) \quad \text{bzw.} \quad F_R{}^+ = (F_{R_1} \cup F_{R_2})^+$$

- Das heißt, jede funktionale Abhängigkeit soll mindestens einer der Teilrelationen vollständig zugeordnet sein (also alle Attribute der linken Seite und das Attribut der rechten Seite müssen in einer der Teilrelationen enthalten sein).
- Zwar führt die Verletzung der Abhängigkeitserhaltung nicht zu einer Veränderung der speicherbaren Information, aber eine „verlorene“ FD ist nicht mehr überprüfbar, ohne dass der Join durchgeführt wird.
  ➔ Verschlechterung der Situation (Redundanzen)

Kanonische Überdeckung

Die *kanonische Überdeckung* F_c einer Menge F von FDs ist eine minimale Menge von FDs, die dieselben (partiellen und transitiven) Abhängigkeiten wie F beschreiben.

F_c ist kanonische Überdeckung von F, wenn gilt:

- $F_c \equiv F$, d.h.: $F_c^+ = F^+$
- In F_c keine FDs $X \rightarrow Y$, bei denen X oder Y überflüssige Attribute enthalten, d.h., es muss gelten:

  1. $\forall A \in X: \left(F_c - (X \rightarrow Y) \cup \left((X - A) \rightarrow Y\right)\right) \not\equiv F_c$
  2. $\forall B \in Y: \left(F_c - (X \rightarrow Y) \cup \left(X \rightarrow (Y - B)\right)\right) \not\equiv F_c$

- Jede linke Seite einer FD ist einzigartig.
  - Dies ist erreichbar durch Anwendung der Vereinigungsregel.

Synthesealgorithmus für 3NF

Synthesealgorithmus für 3NF

Der sogenannte *Synthesealgorithmus* ermittelt zu einem gegebenen Relationenschema R mit funktionalen Abhängigkeiten F eine Zerlegung in Relationen $R_1, \ldots, R_n$ die folgende Kriterien erfüllt:

- $R_1, \ldots, R_n$ ist eine verlustlose Zerlegung von R.
- Die Zerlegung ist abhängigkeitserhaltend.
- Alle R_i $(1 \leq i \leq n)$ sind in dritter Normalform.

Synthesealgorithmus für 3NF

Der Synthese-Algorithmus arbeitet in 4 Schritten:

1. Bestimme die kanonische Überdeckung F_c zu F
2. Erzeuge neue Relationenschemata aus F_c
3. Rekonstruiere einen Schlüsselkandidaten
4. Eliminiere überflüssige Relationen

Synthesealgorithmus für 3NF

1. Bestimme die **kanonische Überdeckung** F_c zu einer gegebenen Menge F von funktionalen Abhängigkeiten (FDs):

a) Führe für jede FD $X \rightarrow Y \in F$ die Linksreduktion durch, also:

- Überprüfe für alle $A \in X$, ob A überflüssig ist, d.h. ob
  $$Y \subseteq AttrHülle(F, X - A)$$
  gilt. Falls dies der Fall ist, ersetze $X \rightarrow Y$ durch $(X - A) \rightarrow Y$.

b) Führe für jede (verbliebene) FD $X \rightarrow Y$ die Rechtsreduktion durch, also:

- Überprüfe für alle B $\in Y$, ob
  $$B \subseteq AttrHülle(F - (X \rightarrow Y) \cup (X \rightarrow (Y - B)), X)$$
  gilt. In diesem Fall ist B auf der rechten Seite überflüssig und kann eliminiert werden, d.h. $X \rightarrow Y$ wird durch $X \rightarrow (Y - B)$ ersetzt.

c) Entferne die FDs der Form $X \rightarrow \{\}$, die in (b) möglicherweise entstanden sind.

d) Fasse FDs der Form $X \rightarrow Y_1, \ldots, X \rightarrow Y_n$ zusammen,
so dass $X \rightarrow (Y_1 \cup \ldots \cup Y_n)$ verbleibt.

Synthesealgorithmus für 3NF

2. Für jede FD $X \rightarrow Y \in F_c$
   - Erzeuge ein Relationenschema $R_X := X \cup Y$.
   - Ordne R_X die FDs $F_X := \{X' \rightarrow Y' \in F_c \mid X' \cup Y' \in R_X\}$ zu.

3. Rekonstruiere einen Schlüsselkandidaten
   - Falls eines der in Schritt 2 erzeugten Schemata einen Schlüsselkandidaten von R bzgl. F_c enthält, sind wir fertig.
   - Andernfalls wähle einen Schlüsselkandidaten $S \subseteq R$ aus und definiere folgendes zusätzliche Schema:
     - $R_S := S$
     - $F_S := \{\}$

4. Eliminiere überflüssige Relationen
   - Eliminiere diejenigen Schemata R_X, die in einem anderen Relationenschema $R_{X'}$ enthalten sind, d.h.
     - $R_X \subseteq R_{X'}$

Synthesealgorithmus für 3NF

Beispiel:

Einkauf (Anbieter, Ware, WGruppe, Kunde, KOrt, KLand, Kaufdatum)

Es gelten folgende FDs: F = {

Kunde, Ware → KLand
Kunde, WGruppe → Anbieter
Anbieter → WGruppe
Ware → WGruppe
Kunde → KOrt
KOrt → KLand

}

Schritte des Synthesealgorithmus:

1. Kanonische Überdeckung F_c der funktionalen Abhängigkeiten:
   a) Linksreduktion:
      Kunde, ~~Ware~~ → KLand,
      {KLand} ⊆ *AttrHülle*(*F*, {*Kunde*, *Ware*}–{*Ware*})
      weil KLand von Kunde alleine (transitiv) funktional abhängig ist
   b) Rechtsreduktion:
      Kunde → ~~KLand~~,
      {KLand} ⊆ *AttrHülle*(*F* – (Kunde → KLand) ∪ (Kunde → {}), Kunde)
      weil KLand transitiv von Kunde funktional abhängig ist
   c) Kunde → {} wird eliminiert
   d) nichts zu tun.

Synthesealgorithmus für 3NF

Beispiel:

Einkauf (Anbieter, Ware, WGruppe, Kunde, KOrt, KLand, Kaufdatum)

Schritte des Synthesealgorithmus:

1. Kanonische Überdeckung F_c der funktionalen Abhängigkeiten:
   Kunde, WGruppe $\rightarrow$ Anbieter
   Anbieter $\rightarrow$ WGruppe
   Ware $\rightarrow$ WGruppe
   Kunde $\rightarrow$ KOrt
   KOrt $\rightarrow$ KLand
2. Erzeugen der neuen Relationenschemata und ihrer FDs:

Bezugsquelle (Kunde, WGruppe, Anbieter)	{Kunde,WGruppe $\rightarrow$ Anbieter, Anbieter $\rightarrow$ WGruppe}
Lieferant (Anbieter, WGruppe)	{Anbieter $\rightarrow$ WGruppe}
Produkt (Ware, WGruppe)	{Ware $\rightarrow$ WGruppe}
Adresse (Kunde, KOrt)	{Kunde $\rightarrow$ KOrt}
Land (KOrt, KLand)	{KOrt $\rightarrow$ KLand}

3. Da keine dieser Relationen einen Schlüsselkandidaten der ursprünglichen Relation enthält, muss noch eine eigene Relation mit dem ursprünglichen Schlüssel angelegt werden:
   Einkauf (Ware, Kunde, Kaufdatum)
4. Da die Relation *Lieferant* in *Bezugsquelle* enthalten ist, können wir *Lieferant* wieder streichen.

Boyce-Codd-Normalform

- Welche Abhängigkeiten können in der dritten Normalform noch auftreten?

  Abhängigkeiten unter Attributen, die prim sind,
  aber noch nicht vollständig einen Schlüssel bilden

- Beispiel:
  Autoverzeichnis (Hersteller, HerstellerNr, ModellNr)
  - es gilt 1:1-Beziehung zw. Hersteller und HerstellerNr:
    Hersteller → HerstellerNr
    HerstellerNr → Hersteller
  - Schlüsselkandidaten sind deshalb:
    {Hersteller, ModellNr}
    {HerstellerNr, ModellNr}
- Schema in 3. NF, da alle Attribute prim sind.

Boyce-Codd-Normalform

- Trotzdem können auch hier Anomalien auftreten

- *Definition:*
  Ein Schema R ist in Boyce-Codd-Normalform, wenn für alle nicht-trivialen Abhängigkeiten $X \rightarrow Y$ gilt:
  - X enthält einen Schlüsselkandidaten von R

- Verlustlose Zerlegung ist generell immer möglich.
- Abhängigkeitserhaltende Zerlegung nicht immer möglich.
  - Falls abhängigkeitserhaltende Zerlegung nicht möglich ist, bevorzugt man die 3NF.

Boyce-Codd-Normalform

Beispiel für nicht-abhängigkeitserhaltende Zerlegung:

Bank (Filiale, Kunde, Betreuer)

FDs: F = { Betreuer → Filiale

Kunde, Filiale → Betreuer }

- Schema nicht in BCNF, da das Attribut Betreuer keinen Schlüssel enthält.
- Zerlegung in BCNF:

  Personal (Filiale, Betreuer)

  Kunden (Kunde, Betreuer)

  – ist verlustlos: Personal ⋈ Kunden = Bank

  (da: Betreuer → Betreuer, Filiale)

  – nicht abhängigkeitserhaltend, da

  (Kunde, Filiale → Betreuer) verlorengegangen ist

Mehrwertige Abhängigkeiten (MVD)

- Mehrwertige Abhängigkeiten entstehen, wenn mehrere unabhängige 1:*n*-Beziehungen in einer Relation stehen (was nach Kapitel 5 eigentlich nicht sein darf):
- **Mitarbeiter** (Name, Projekte, Verwandte)

Name	Projekte	Verwandte
Huber,	{P1, P2, P3}	{Heinz, Hans, Hubert}
Müller,	{P2, P3}	{Manfred}

- In erster Normalform müsste man mindestens 3 Tupel für Huber und 2 Tupel für Müller speichern:
- **Mitarbeiter** (Name, Projekte, Verwandte)

Name	Projekte	Verwandte
Huber,	P1,	Heinz,
Huber,	P2,	Hans,
Huber,	P3,	Hubert,
Müller,	P2,	Manfred
Müller,	P3,	NULL

Mehrwertige Abhängigkeiten (MVD)

- Um die Anfrage zu ermöglichen, wer die Verwandten von Mitarbeitern in Projekt P2 sind, müssen pro Mitarbeiter sogar sämtliche Kombinationstupel gespeichert werden:

Mitarbeiter(Name, Projekte, Verwandte)

Name	Projekte	Verwandte
Huber,	P1,	Heinz,
Huber,	P1,	Hans,
Huber,	P1,	Hubert,
Huber,	P2,	Heinz,
Huber,	P2,	Hans,
Huber,	P2,	Hubert,
Huber,	P3,	Heinz,
Huber,	P3,	Hans,
Huber,	P3,	Hubert,
Müller,	P2,	Manfred,
Müller,	P3,	Manfred.

- Wir nennen dies eine *Mehrwertige Abhängigkeit* (engl. Multivalued Dependency, MVD)

Mehrwertige Abhängigkeiten (MVD)

Gegeben: $X,Y \subseteq R$, und es sei $Z = R \setminus (X \cup Y)$ (d.h. der Rest)

Y ist ***mehrwertig abhängig*** von X ($X \twoheadrightarrow Y$), wenn für jede gültige Ausprägung von R gilt:

Für jedes Paar aus Tupeln t_1, t_2 mit $t_1.X=t_2.X$, aber $t_1 \neq t_2$, existieren die (nicht immer zu t_1 und t_2 verschiedenen) Tupel t_3 und t_4 mit den Eigenschaften:

$$t_1.X = t_2.X = t_3.X = t_4.X$$
$$t_3.Y = t_1.Y$$
$$t_3.Z = t_2.Z$$
$$t_4.Y = t_2.Y$$
$$t_4.Z = t_1.Z$$

D.h. jedem X ist eine Menge von Y-Werten zugeordnet.

Jede FD ist auch MVD

(diese Menge ist dann jeweils ein-elementig und $t_3=t_2$, $t_4=t_1$)

Beispiel MVD

	R		
	X $\overbrace{A_1 \ldots A_i}$	**Y** $\overbrace{A_{1+1} \ldots A_j}$	**Z** $\overbrace{A_{j+1} \ldots A_n}$
t_1	$a_1 \ldots a_i$	$a_{i+1} \ldots a_j$	$a_{j+1} \ldots a_n$
t_2	$a_1 \ldots a_i$	$b_{i+1} \ldots b_j$	$b_{j+1} \ldots b_n$
t_3	$a_1 \ldots a_i$	$a_{i+1} \ldots a_j$	$b_{j+1} \ldots b_n$
t_4	$a_1 \ldots a_i$	$b_{i+1} \ldots b_j$	$a_{j+1} \ldots a_n$

Weiteres Beispiel

Relation: Modelle

ModellNr	Farbe	Leistung
E36	blau	170 PS
E36	schwarz	198 PS
E36	blau	198 PS
E36	schwarz	170 PS
E34	schwarz	170 PS

$\{ModellNr\} \rightarrow\rightarrow \{Farbe\}$ und $\{ModellNr\} \rightarrow\rightarrow \{Leistung\}$

Farben

ModellNr	Farbe
E36	blau
E36	schwarz
E34	Schwarz

Leistung

ModellNr	Leistung
E36	170 PS
E36	198 PS
E34	170 PS

$Modelle = \Pi_{ModellNr,Sprache}(Farben) \bowtie \Pi_{ModellNr,Leistung}(Leistung)$

Verlustlose Zerlegung MVD

Ein Relationenschema R mit einer Menge D von zugeordneten funktionalen mehrwertigen Abhängigkeiten kann genau dann verlustlos in die beiden Schemata R_1 und R_2 zerlegt werden, wenn gilt:

- $R = R_1 \cup R_2$
- mindestens eine von zwei MVDs gilt:
  1. $R_1 \cap R_2 \rightarrow\rightarrow R_1$ oder
  2. $R_1 \cap R_2 \rightarrow\rightarrow R_2$

Triviale MVD und 4. Normalform

Eine MVD $X \rightarrow\rightarrow Y$ bezogen auf $R \supseteq X \cup Y$ ist *trivial*, wenn jede mögliche Ausprägung r von R diese MVD erfüllt.

Man kann zeigen, dass $X \rightarrow\rightarrow Y$ trivial ist, genau dann wenn:

- $Y \subseteq X$ oder
- $Y = R - X$.

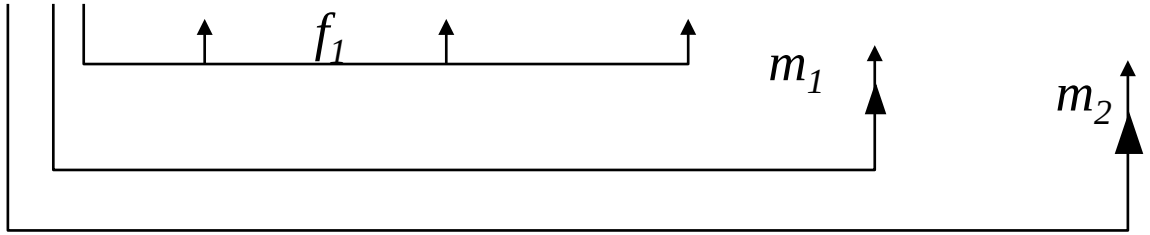
Eine Relation R mit zugeordneter Menge F von funktionalen und mehrwertigen Abhängigkeiten ist in 4. Normalform (**4NF**), wenn für jede MVD $X \rightarrow\rightarrow Y \in F^+$ eine der folgenden Bedingungen gilt:

- Die MVD ist trivial oder
- X ist ein Superschlüssel von R.

Beispiel

Assistenten

(PersNr,Name,Fachgebiet, Boss,Sprache,ProgSprache)



- **Assistenten** (PersNr,Name,Fachgebiet, Boss)
- **Sprachen** (PersNr, Sprache)
- **ProgSprach** (PersNr, ProgSprache)

Schlussbemerkungen

- Extrem-Ansatz: Universal Relation Assumption:
  - Modelliere alles zunächst in einer Tabelle
  - Ermittle die funktionalen Abhängigkeiten
  - Zerlege das Relationenschema entsprechend (der letzte Schritt kann auch automatisiert werden: Synthesealgorithmus für die 3. Normalform)
- Ein gut durchdachtes E/R-Diagramm liefert bereits weitgehend normalisierte Tabellen.

Schlussbemerkungen

- Normalisierung kann schädlich für die Performanz sein, weil Joins sehr teuer auszuwerten sind.
- Nicht *jede* FD berücksichtigen:
  - Abhängigkeiten zw. Wohnort, Vorwahl, Postleitzahl
  - Man kann SQL-Integritätsbedingungen formulieren, um Anomalien zu vermeiden (sog. Trigger).
- Aber es gibt auch Konzepte, Relationen so abzuspeichern, dass Join auf bestimmten Attributen unterstützt wird
  - ORACLE-Cluster

Zusammenfassung

Implikation

- 1. Normalform:
  Alle Attribute atomar
- 2. Normalform:
  Keine funktionale Abhängigkeit eines Nicht-Schlüssel-Attributs von Teil eines Schlüssels
- 3. Normalform:
  Zusätzlich keine nicht-triviale funktionale Abhängigkeit eines Nicht-Schlüssel-Attributs von Nicht-Schlüssel-Attributen
- Boyce-Codd-Normalform:
  Zusätzlich keine nicht-triviale funktionale Abhängigkeit unter den Schlüssel-Attributen
- 4. Normalform:
  keine Redundanz durch MVDs.